Docket No. 219053US2SRD

#2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Mikito IWAMASA

GAU: 04/25/02

SERIAL NO: New Application

EXAMINER:

FILED: Herewith

FOR: METHOD AND COMPUTER PROGRAM PRODUCT FOR SYSTEM DESIGN SUPPORT

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.

☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
JAPAN	2001-024491	January 31, 2001

Certified copies of the corresponding Convention Application(s)

☒ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

James D. Hamilton
Registration No. 28,421

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

01S1545

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　１月３１日

出願番号
Application Number: 特願２００１－０２４４９１

出願人
Applicant(s): 株式会社東芝

２００１年１２月１４日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３１０８８９５

【書類名】　特許願
【整理番号】　A000100060
【提出日】　平成13年 1月31日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 15/00
【発明の名称】　設計支援装置及び設計支援方法並びにシステム設計支援装置
【請求項の数】　18
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　岩政　幹人
【特許出願人】
【識別番号】　000003078
【氏名又は名称】　株式会社　東芝
【代理人】
【識別番号】　100058479
【弁理士】
【氏名又は名称】　鈴江　武彦
【電話番号】　03-3502-3181
【選任した代理人】
【識別番号】　100084618
【弁理士】
【氏名又は名称】　村松　貞男
【選任した代理人】
【識別番号】　100068814
【弁理士】
【氏名又は名称】　坪井　淳

出証特２００１－３１０８８９５

【選任した代理人】

【識別番号】　100092196

【弁理士】

【氏名又は名称】　橋本　良郎

【選任した代理人】

【識別番号】　100091351

【弁理士】

【氏名又は名称】　河野　哲

【選任した代理人】

【識別番号】　100088683

【弁理士】

【氏名又は名称】　中村　誠

【選任した代理人】

【識別番号】　100070437

【弁理士】

【氏名又は名称】　河井　将次

【手数料の表示】

【予納台帳番号】　011567

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　設計支援装置及び設計支援方法並びにシステム設計支援装置

【特許請求の範囲】

【請求項１】

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム生成手段を備えたことを特徴とする設計支援装置。

【請求項２】

前記状態遷移に関する情報は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含むものであることを特徴とする請求項１に記載の設計支援装置。

【請求項３】

前記状態遷移に関する情報は、状態が遷移できるための条件を示す情報と、次状態に遷移する前に実行すべきアクションを示す情報との少なくとも一方をさらに含むものであることを特徴とする請求項２に記載の設計支援装置。

【請求項４】

前記実行制御に関する情報は、前記状態遷移に関連して実行制御が惹起される実行処理内容を示す情報及び該実行処理内容に対する実行制御の種類を示す情報を含み、

前記実行制御の種類は、少なくとも、処理の起動、イベント発生に起因する割込による処理の強制終了、イベント発生に起因する割込による処理の一時停止、処理が終了したことを知らせるイベントの通知、割込による処理の一時停止からの復帰を含むことを特徴とする請求項１に記載の設計支援装置。

【請求項５】

前記システム生成手段は、前記システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシス

テム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する実行制御表変換手段を含むことを特徴とする請求項１ないし４のいずれか１項に記載の設計支援装置。

【請求項６】

前記実行状態の遷移に関する情報は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むものであることを特徴とする請求項５に記載の設計支援装置。

【請求項７】

前記システム生成手段は、前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム実装変換手段を更に含むことを特徴とする請求項５または６に記載の設計支援装置。

【請求項８】

前記システム実装変換手段は、個々の前記実行状態の遷移に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開する手段と、展開された個々のシステム記述言語で記述された仕様を所定の仕様統合ルールに基づいて統合する手段とを含むことを特徴とする請求項７に記載の設計支援装置。

【請求項９】

生成された前記システム記述言語で記述されたシステム仕様を実行するための手段を更に備えたことを特徴とする請求項１ないし８のいずれか１項に記載の設計支援装置。

【請求項１０】

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成し、

生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語

で記述された実行可能な形式を持つシステム仕様を生成することを特徴とする設計支援方法。

【請求項１１】

前記状態遷移表形式で記述されたシステム仕様は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含む状態遷移単位情報の集合からなるものであることを特徴とする請求項１０に記載の設計支援方法。

【請求項１２】

前記実行制御に関する情報は、前記状態遷移に関連して実行制御が惹起される実行処理内容を示す情報及び該実行処理内容に対する実行制御の種類を示す情報を含み、

前記実行制御の種類は、少なくとも、処理の起動、イベント発生に起因する割込による処理の強制終了、イベント発生に起因する割込による処理の一時停止、処理が終了したことを知らせるイベントの通知、割込による処理の一時停止からの復帰を含むことを特徴とする請求項１１に記載の設計支援方法。

【請求項１３】

前記実行制御表形式のシステム仕様は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むプログラム状態遷移単位情報の集合からなるものであることを特徴とする請求項１１または１２に記載の設計支援方法。

【請求項１４】

前記実行制御表形式のシステム仕様を生成するに際しては、前記状態遷移単位情報の集合に基づいて前記プログラム状態遷移単位情報の集合を生成し、

前記システム記述言語で記述されたシステム仕様を生成するに際しては、個々の前記プログラム状態遷移単位情報に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開し、展開された個々のシステム記述言語で記述された仕様を所定の仕様統合ルールに基づいて統合することを特徴とす

る請求項１３に記載の設計支援方法。

【請求項１５】

設計支援装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する機能と、

生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成する機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体。

【請求項１６】

設計支援装置としてコンピュータを機能させるためのプログラムであって、

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する機能と、

生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成する機能とをコンピュータに実現させるためのプログラム。

【請求項１７】

計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、

特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てるためのアーキテクチャ探索手段と、

特定のアーキテクチャに対し仕様要素が割り当てられたシステム仕様モデルに

つき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、

システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、

システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、

前記仕様モデル記述手段は、請求項１ないし９のいずれか１項に記載の設計支援装置に相当する機能を含むものであることを特徴とするシステム設計支援装置。

【請求項１８】

前記仕様モデル記述手段、前記アーキテクチャ探索手段または前記コミュニケーション合成手段の少なくとも一つにおいて、システム仕様モデルを部品化し、これを設計に再利用するための部品化・再利用手段を更に備えたことを特徴とする請求項１７に記載のシステム設計支援装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、計算機や電子情報機器における、ハードウェア、ソフトウェア、及びこれら両方を含むシステムの設計支援を行うための設計支援装置及び設計支援方法並びにシステム設計支援装置に関する。

【０００２】

【従来の技術】

計算機システムにおいては、ＵＭＬ（Ｕｎｉｆｉｅｄ　Ｍｏｄｅｌｉｎｇ　Ｌａｎｇｕａｇｅ）等のオブジェクト指向手法に代表されるように、システム分析の手法として状態遷移形式が用いられてきた（なお、ＵＭＬは例えば「UMLｶﾞｲﾄﾞﾌﾞｯｸ」,Hans-Eriksson/Magnus Penker著,杉本宣男/落合修/武田多美子 監訳,ISBN4-8101-8987-2に詳しい）。状態遷移形式は、システムに含まれる複数の状態を抽出し、それら状態が、イベントや条件によって、どのように互いに遷移するかを整理したものである。そして、システム分析が終わった後に、状態遷移形式を参

考にしながら、システム設計を開始するという手順を踏んでいた。

【０００３】

一方、システム開発期間の短縮、および設計品質の向上という観点からは、分析をしながら仕様を実際に実行して確かめる、いわゆるラピッドプロトタイピングの機能や、システム分析とシステム設計のシームレスな結合機能が必要不可欠である。

【０００４】

従来の技術では、以上の事情に鑑み、状態遷移による分析結果に基づくシミュレーションを実行したり、実行プログラムに変換する手法が実現されてきた（ＢｅｔｔｅｒＳｔａｔｅ）。これらの技術では、状態遷移の瞬間に実行されるアクション、及びシステムが状態に滞留するときに実行されるアクティビティに、実行プログラムを割り当てることにより、シミュレーション機能や変換機能を実現している。

【０００５】

【発明が解決しようとする課題】

状態遷移形式では、アクションは瞬時に実行が終了することが前提となっており、０より大きい一定の実行時間を必要とする計算機プログラムに代表される実行処理内容を割り当てることは、意味論的に不整合であり、システム全体の挙動の整合性が保証できない。

【０００６】

また、アクティビティは状態と処理が一対一に対応しているので、遷移が異なれば遷移先や遷移元が同じ状態でも異なるアクションが実行されるという状態遷移形式の記述の柔軟性が損なわれ、記述できる範囲が限定されてしまうという問題がある。この結果、状態遷移形式のアクションやアクティビティに計算機プログラム等の実行処理内容を割り当てる方式では、精度のよいシミュレーションが実行できなかったり、また柔軟なシステム設計ができないという課題があった。

【０００７】

本発明は、上記事情を考慮してなされたもので、システム分析から設計までを一環して支援し、システム分析結果に対応した精度良いシステム設計の支援を行

って、システム開発期間を短縮させ、設計品質を向上させることの可能な設計支援装置及び設計支援方法並びにシステム設計支援装置を提供することを目的とする。

【０００８】

【課題を解決するための手段】

本発明に係る設計支援装置は、システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システム記述言語で記述された（例えば、計算機や電子部品組込み機器によって）実行可能な形式を持つシステム仕様を生成するシステム生成手段を備えたことを特徴とする。

【０００９】

好ましくは、前記状態遷移に関する情報は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含むものであるようにしてもよい。

【００１０】

好ましくは、前記システム生成手段は、前記システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する実行制御表変換手段を含むようにしてもよい。好ましくは、前記実行状態の遷移に関する情報は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むものであるようにしてもよい。好ましくは、前記システム生成手段は、前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム実装変換手段を更に含むようにしてもよい。

【００１１】

本発明に係る設計支援方法は、システムの状態遷移に関する情報に対し、関連

を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成し、生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成することを特徴とする。

【００１２】

好ましくは、前記状態遷移表形式で記述されたシステム仕様は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含む状態遷移単位情報の集合からなるものであるようにしてもよい。好ましくは、前記実行制御表形式のシステム仕様は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むプログラム状態遷移単位情報の集合からなるものであるようにしてもよい。好ましくは、前記実行制御表形式のシステム仕様を生成するに際しては、前記状態遷移単位情報の集合に基づいて前記プログラム状態遷移単位情報の集合を生成し、前記システム記述言語で記述されたシステム仕様を生成するに際しては、個々の前記プログラム状態遷移単位情報に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開し、展開された個々のシステム記述言語で記述された仕様を所定の仕様統合ルールに基づいて統合するようにしてもよい。

【００１３】

また、本発明に係るシステム設計支援装置は、計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てるためのアーキテクチャ探索手段と、特定のアーキテクチャに対し仕様要

素が割り当てられたシステム仕様モデルにつき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、前記仕様モデル記述手段は、請求項１ないし９のいずれか１項に記載の設計支援装置に相当する機能を含むものであることを特徴とするシステム設計支援装置。

【００１４】

なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させるための）プログラムとしても成立し、該プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

【００１５】

本発明によれば、状態遷移形式の柔軟さを保持しつつ全体のシステムとしての整合性も保証する方式で、実行処理内容を遷移状態形式に割り当てて、この状態遷移形式から、実行可能なシステムを生成する手段を具備する設計支援装置を提供することができる。

【００１６】

そして、本発明によれば、状態遷移表形式に処理内容（プログラム）を直接入力して、これを実行可能なシステムに変換するシステム生成手段を具備しているので、状態遷移表の柔軟さを損なずにシステム設計を行うことができ、また処理内容の実行制御命令自体は瞬間に実行されるので意味論的に状態遷移の仕組みと整合性が取れたシステム設計を行うことができる。

【００１７】

従って、本発明によれば、システム分析から設計までを一環して支援し、システム分析結果に対応した精度よいシステム設計の支援を行って、システム開発期

間を短縮させ、設計品質を向上させることができる。

【００１８】

【発明の実施の形態】

以下、図面を参照しながら発明の実施の形態を説明する。

【００１９】

図１１に、本発明の実施の形態に係るシステム設計支援装置１００の構成例を示す。図１１に示されるように、システム設計支援装置１００は、仕様モデル記述部１０１、アーキテクチャ探索部１０２、コミュニケーション合成部１０３、ハードウェア仕様生成部１０４、部品化・再利用部１０５、ソフトウェア仕様生成部１０６、システム仕様記録部１０７を備えている。システム設計支援装置１００は、例えば、コンピュータを用いて構成可能であり、この場合、各処理部分はプログラムによって実施できる。

【００２０】

システム仕様記録部１０７は、入力データや各処理部の処理結果の全部又は一部、例えば、計算機で実行されるソフトウェアの仕様や、半導体を組み合わせたハードウェアの仕様や、ソフトウェアとハードウェアを組み合わせた組み込みシステムの仕様や、ワークフロー等のビジネスプロセスの仕様に対しシステムレベルでの仕様等や、その他の必要なデータを記録するためのものである。

【００２１】

仕様モデル記述部１０１は、設計者が予め定められた仕様記述形式に従って計算内容の仕様、通信内容の仕様を記述することを支援するためのものである。仕様モデル記述部１０１の支援の結果、仕様モデルが生成される。仕様記述形式には、例えば、構造化プログラミング言語に代表される構造化テキスト形式や、グラフを利用する構造図形式や、テーブルを利用する表形式が有る。

【００２２】

アーキテクチャ探索部１０２は、与えられた仕様記述モデルと、アーキテクチャ（例えば、ハードウェア実行環境とソフトウェア実行環境の構成）に対し、仕様の内容を保存したまま仕様モデルの部分構造を分割しアーキテクチャ要素に分配する。すなわち、仕様モデル記述部１０１で設計された計算内容の仕様や通信

内容の仕様を構成する部品をアーキテクチャ要素に割り当てる。

【００２３】

コミュニケーション合成部１０３は、アーキテクチャ探索部１０２で分配された通信仕様要素間に対して通信手順（プロトコル）を挿入し、通信内容の仕様が挿入された通信手順と整合性を保つためのプロトコル変換（通信手順の組替え）を行う。

【００２４】

なお、アーキテクチャ探索部１０２が処理対象とする仕様記述モデルは、仕様モデル記述部１０１を利用して記述されたものに限定されず、他のツール等で記述されたものでもよい。また、コミュニケーション合成部１０３が処理対象とする仕様記述モデルは、アーキテクチャ探索部１０２の出力結果（またはこの結果に修正を加えたもの）に限定されず、他のツール等で記述されたものでもよい。

【００２５】

ハードウェア仕様生成部１０４は、システム仕様からハードウェア記述言語などによるハードウェア仕様を生成する。

【００２６】

ソフトウェア仕様生成部１０６は、システム仕様からソフトウェア記述言語などによるソフトウェア仕様を生成する。

【００２７】

ハードウェア仕様生成部１０４やソフトウェア仕様生成部１０６は、仕様モデル記述部１０１の出力結果（またはこの結果に修正を加えたもの）、アーキテクチャ探索部１０２の出力結果（またはこの結果に修正を加えたもの）、コミュニケーション合成部１０３の出力結果（またはこの結果に修正を加えたもの）、あるいは他のツール等で記述されたものなどを処理対象とすることができる。

【００２８】

部品化・再利用部１０５は、仕様モデル記述部１０１、アーキテクチャ探索部１０２、コミュニケーション合成部１０３の全部または一部の設計段階において、仕様を部品化し、これを設計に再利用するためのものである。

【００２９】

以下詳述する設計支援装置は、図１１のシステム設計支援装置の仕様モデル記述部１０１に用いられるもので、状態に基づくシステム分析からプログラムのシステム設計をシームレスに実行できることを可能にするものである。

【００３０】

図１に、本発明の一実施形態に係る設計支援装置の構成例を示す。

【００３１】

図１に示されるように、本設計支援装置は、実行制御表変換部１１およびシステム実装変換部１２を含むシステム生成部１、外部とデータや指示のやり取りを行うための入出力部２、入力となる状態遷移表形式で記述された仕様（２１）、中間状態となる実行制御表形式で記述された仕様（２２）及び出力となるシステム記述言語で記述された仕様（２３）、並びに変換ルール及び仕様統合ルールなどを記憶するための記憶部（図示せず）を備えている。また、必要に応じて、通信部３やプログラム実行部４などを更に備えることができる。

【００３２】

システム実装変換部１２の実行制御表変換部１１は、状態遷移表形式で記述された仕様（２１）を、実行制御表形式で記述された仕様（２２）に変換する。

【００３３】

システム実装変換部１２のシステム実装変換部１２は、実行制御表形式で記述された仕様（２２）を、システム記述言語で記述された仕様（２３）に変換する。

【００３４】

本設計支援装置は、例えば、コンピュータ上で本発明を実施したプログラム（設計支援ソフト）を実行することによって、実現可能である。

【００３５】

以下では、まず「状態遷移表形式で記述された仕様（状態遷移表形式の仕様）」と「実行制御表形式で記述された仕様（実行制御表形式の仕様）」の意味内容を説明した上で、前者を後者に変換する実行制御表変換部１１の処理について説明する。

【００３６】

＜状態遷移表形式の仕様＞

遷移状態形式の仕様とは、システムの挙動を、システムが取り得る有限個の状態と「イベント」による状態間の遷移の定義により記述する仕組みである（図４あるいは図７の具体例参照）。

【００３７】

状態遷移表形式の仕様は、例えば下記に示す状態遷移単位の集合によって構成される。

状態遷移単位＝（遷移番号、現状態、イベント、遷移条件、
アクション、次状態、実行制御）

なお、状態遷移単位＝（遷移番号、現状態、イベント、遷移条件、次状態、実行制御）、状態遷移単位＝（遷移番号、現状態、イベント、アクション、次状態、実行制御）、状態遷移単位＝（遷移番号、現状態、イベント、次状態、実行制御）なども可能である。

【００３８】

遷移番号で識別される個々の状態遷移単位は、システムが「現状態」にあるとき、「イベント」が発生して、かつ「遷移条件」を満たせば、「アクション」を実行して「次状態」に遷移する、というシステムの挙動の一部を表している。このとき、遷移時のアクションは一瞬で終了する。「実行制御」は、遷移が発生したときの、実行処理内容（プログラム）の実行制御に関する項目である。

【００３９】

例えば図４のＣＤプレーヤの具体例（ただしアクションは省かれている）における番号［２］の状態遷移単位は、再生状態（ｐｌａｙｉｎｇ）において、ポーズ（ｐａｕｓｅ）イベントが発生した場合に、ｐｌａｙプログラムに割り込み（ｉｎｔｅｒｒｕｐｔ）をかけて一時停止させるとともに、ｐａｕｓｅプログラムを起動（ｓｔａｒｔ）させ、ポーズ（ｐａｕｓｅ）状態に遷移する、という仕様を表している。

【００４０】

なお、特別な状態として、初期状態（ｓｔａｒｔ）、終了状態（ｅｎｄ）が準備される。

初期状態（ｓｔａｒｔ）は、システムが起動したときの最初の状態を指示するポインタであり、状態遷移単位の一つとして以下のように指示が表現される。

（ｓｔａｒｔ、－、－、－、最初の状態）

終了状態（ｅｎｄ）は、終了したときに遷移する仮想的な状態である。

【００４１】

また、状態遷移表形式は、階層的に記述することもできる。すなわち、下位階層の状態遷移単位の集合を一つの階層状態Ｘにまとめて、上位階層でＸを一つの状態として含む状態遷移表形式の仕様を定義することができる。上位階層において階層状態Ｘに遷移したときは、Ｘの下位階層では初期状態（ｓｔａｒｔ）で指示する状態に遷移する。

【００４２】

＜遷移の接続について＞

状態遷移表形式では、現状態から次状態への「遷移」を定義するが、当該次状態は、次の時間では、現状態になるので、状態遷移単位同士の間には同一の「状態」を介して、方向性のある隣接関係が成立する。この隣接関係が成立する遷移同士は「接続している」と呼ぶことにする。遷移の接続関係は、遷移の方向性に従う半順序関係である（例：「推移律が成り立つ」＝「遷移Ａと遷移Ｂが接続し、遷移Ｂと遷移Ｃが接続するならば、遷移Ａと遷移Ｃは接続関係にある」）。また、遷移の方向に従った遷移接続を順方向の接続、その逆を逆方向の接続と呼ぶことにする。例えば、遷移Ａとが遷移Ｂが接続し、接続の方向が遷移Ａから遷移Ｂである（遷移Ａの後で遷移Ｂが発生する）とき、遷移Ａから遷移Ｂへの接続が順方向になり、遷移Ｂから遷移Ａへの接続が逆方向になる。

【００４３】

＜「実行制御」の細目＞

状態遷移形式における「実行制御」の項目では、プログラムの実行制御に関する細項目を用意して、この細項目に処理内容の単位（プログラム）を入力する。ここで、細項目とは、少なくとも、起動（ｓｔａｒｔ）、停止（ｓｔｏｐ）、割込み一時停止（ｉｎｔｅｒｒｕｐｔ）、プログラムの終了（ｆｉｎｉｓｈ）、一時停止からの復帰（ｒｅｓｕｍｅ）、及びこれらの要素を組み合わせた項目で構

成される。

【００４４】

なお、処理内容の単位（プログラム）が終了したことも、イベントとみなす。これを、特別に、終了イベントと呼び、状態遷移表形式では［ＦＩＮＩＳＨ］と表記するものとする（図４の番号［３］の状態遷移単位参照）。

【００４５】

＜実行制御表形式の仕様＞

実行制御表形式の仕様とは、システムにおけるプログラム等の実行処理内容の単位と、これらの処理内容単位（プログラム）が状態遷移によってどのように実行制御（起動、停止、割込）されるかを定義することにより、システムの挙動を記述する仕組みである（図８の具体例参照）。

【００４６】

実行制御表形式の仕様は、処理内容単位（プログラム）が状態遷移によってどのように切り替わるかを示すプログラム状態遷移単位の集合によって構成される。

【００４７】

プログラム状態遷移単位は、状態遷移に起因する処理内容単位（プログラム）の実行状態（待機状態、実行状態、終了状態、一時停止状態）の変化を、他プログラムの実行状態の変化と関連付けたテーブルである。

【００４８】

プログラム状態遷移単位は、例えば以下にテーブで表現される。

プログラム状態遷移単位＝（番号、現プログラム、遷移、移動タイプ、
終了タイプ、次プログラム）

個々のプログラム状態遷移単位は、「現プログラム」実行中に「遷移」が発生すると、「終了タイプ」に従って「現プログラム」を実行制御し、「起動タイプ」に従って「次プログラム」を起動するという仕様を表している。

【００４９】

例えば図８のＣＤプレーヤの具体例における番号（２）のプログラム状態遷移単位は、現プログラム（ｐｌａｙ）の実行中に遷移［２］が発生すると、現プロ

グラム（ｐｌａｙ）を割込一時停止させ、次現プログラム（ｐａｕｓｅ）を起動させる、という仕様を表している。

【００５０】

起動タイプには、「起動」、「復帰」の２種類がある。

「起動」は、プログラムを起動することを示す。

「復帰」は、一時停止されていたプログラムの実行を再開することを示す。

終了タイプには、「割込終了」、「完了」、「割込一時停止」の３種類がある。

「完了」は、前プログラムが終了したときに次プログラムを起動することを示す。

「割込終了」は、現プログラムが実行中に、遷移が発生すると、現プログラムを強制終了して、次プログラムを起動することを示す。

「割込一時停止」は、現プログラムが実行中に、遷移が発生すると、現プログラムを一時停止して、次プログラムを起動することを示す。

【００５１】

また、例えば、

（現ﾌﾟﾛｸﾞﾗﾑA、遷移Ｘ、移動タイプＹ、終了タイプＺ、次ﾌﾟﾛｸﾞﾗﾑＢ）

（現ﾌﾟﾛｸﾞﾗﾑA、遷移Ｘ、移動タイプＹ、終了タイプＺ、次ﾌﾟﾛｸﾞﾗﾑＣ）

のように、同一の（現プログラム、遷移Ｘ、移動タイプＹ、終了タイプＺ）の組み合わせに対し複数の「次プログラム」が定義されるときには、当該複数の次プログラム（上記の例では、Ｂ，Ｃ）が遷移Ｘにより並列起動されることを表すこととする。

【００５２】

なお、実行制御表形式の仕様でも、状態遷移表形式の仕様と同様に、特別なプログラム状態として、開始状態（ｓｔａｒｔ）と終了状態（ｅｎｄ）がある。

【００５３】

また、実行制御表形式の仕様でも、階層的に記述することもできる。すなわち、下位階層のプログラム状態遷移単位の集合を一つの階層状態Ｘにまとめて、上位階層でＸを実行処理内容（プログラム）の一つとして含むプログラム状態遷移

表形式の仕様を定義することができる。上位階層において階層状態Ｘに遷移したときは、Ｘの下位階層では初期状態（ｓｔａｒｔ）で指示されるプログラム状態が実行される。また、階層状態Ｘに対して「割込終了」「割込一時停止」が行われたときは、Ｘの下位階層全てのプログラムに対して「割込終了」「割込一時停止」が行われる。

【００５４】

＜実行制御表変換部１１の処理＞

続いて、実行制御表変換部１１の処理について説明する。

【００５５】

実行制御表変換部１１は、状態遷移表形式における実行制御の項目（ｓｔａｒｔ，ｓｔｏｐ，ｉｎｔｅｒｒｕｐｔ，ｒｅｓｕｍｅ，ｆｉｎｉｓｈ）の情報に基づき、状態遷移表を実行制御表形に変換する。

【００５６】

図２に、実行制御表変換部１１の処理手順の一例を示す。

【００５７】

<<ステップＳ１：プログラム状態遷移単位への展開>>

状態遷移表における個々の状態遷移単位に対し、実行制御の開始に関する項目（ｓｔａｒｔ，ｒｅｓｕｍｅ）から一つプログラムを選択して、これを「現プログラム」とするとともに、実行制御の終了に関する項目（ｓｔｏｐ，ｆｉｎｉｓｈ，ｉｎｔｅｒｒｕｐｔ）から一つプログラムを選択して、これを「次プログラム」として得られる、＜現プログラム、次プログラム＞の組み合わせを展開する。

【００５８】

現プログラムの由来となる実行制御項目（ｓｔａｒｔ，ｒｅｓｕｍｅ）に応じて、次のように「起動タイプ」を決定する。

ｓｔａｒｔ　ならば、「起動タイプ」＝起動

ｒｅｓｕｍｅならば、「起動タイプ」＝復帰

また、次プログラムの由来となる実行制御項目（ＦＩＮＩＳＨ，ｉｎｔｅｒｒｕｐｔ，ｓｔｏｐ）に応じて、次のように「終了タイプ」を決定する。

ｓｔｏｐ　　ならば、「起動タイプ」＝割込終了

ｆｉｎｉｓｈならば、「起動タイプ」＝完了

interrupt　ならば、「起動タイプ」＝割込一時停止

また、「遷移」には、遷移番号を割り当てる。

【００５９】

１つの状態遷移単位に対して、１つ以上のプログラム状態遷移単位が展開される。

【００６０】

以上の処理を状態遷移表に含まれる全ての状態遷移単位に対して行うことにより、プログラム状態遷移単位の集合すなわち実行制御表を得る。

【００６１】

<<ステップＳ２：遷移を検索し、プログラム状態遷移単位を生成する>>

状態遷移表形式を実行制御表形式に変換した時点において、プログラム状態単位、

（現プログラム、遷移、起動タイプ、終了タイプ、次プログラム）

の中に、現プログラムや次プログラムが無いものがある場合は、状態遷移表における遷移の接続関係を参照して、「遷移」に対して、

「次プログラム」項目が無い場合は、順方向、

「現プログラム」項目が無い場合は、逆方向、

となる遷移番号を「遷移」に持つプログラム状態遷移単位を検索し、「次プログラム」、「現プログラム」に相当する処理単位（プログラム）に突き当たるまで、これを繰り返す。

【００６２】

プログラムが見つかったときは、見つかったプログラムを各々「次プログラム」、「現プログラム」とする新たなプログラム状態遷移単位を作成し、検索を行った過程のプログラム状態遷移単位（複数）を削除する。このとき、新たに作成したプログラム状態遷移単位の「遷移」は、検索を行った全ての遷移の遷移列とする。

【００６３】

<<ステップＳ３：新しい処理内容の追加>>

見つかったプログラムと元々のプログラムとの間を介在する新しい実行処理（プログラム）を追加する。遷移の接続性を考慮して、起動・終了部分に適切に挿入することで実現される。

【００６４】

以上の手順に従い、状態遷移表が実行制御表に変換される。

【００６５】

＜状態遷移表形式と実行制御形式の違い＞

状態遷移表形式は、イベントに応じた状態の変化により、システムの挙動を記述する。システム分析過程では、重要な記述である。システム設計の段階においては、システム分析の結果を踏まえ、今度は状態ではなく、計算機プログラム等の実行処理内容を同定し、どのタイミングで、どの実行処理内容を起動したり停止したりするかに関する制御方式を設計する。

【００６６】

状態遷移表形式でも、ＳｔａｔｅＣｈａｒｔ方式に代表されるように、遷移時に実行されるアクションや、状態滞留中に実行されるアクティビティとして、処理内容を状態遷移に記述する方法もあるが、状態遷移形式ではアクションは瞬時に実行が終了することが前提となっており、０より大きい一定の実行時間を必要とする計算機プログラムに代表される実行処理内容を割り当てることは意味論的に不整合であり、システム全体の挙動の整合性が保証できない。また、アクティビティは状態と処理が一対一に対応しているので、遷移が異なれば遷移先や遷移元が同じ状態でも異なるアクションが実行されるという状態遷移形式の記述の柔軟性が損なわれ、記述できる範囲が限定されてしまうという問題があり、結果として状態遷移形式のアクションやアクティビティに計算機プログラム等の実行処理内容を割り当てる方式では、精度のよいシミュレーションが実行できなかったり、また柔軟なシステム設計ができないという課題がある。

そこで、状態遷移ではなく、処理内容の実行制御を定義する新しい方式が、実行制御表形式である。

【００６７】

次に、実行制御表形式で記述された仕様を、システム記述言語で記述された仕様に変換する、システム実装変換部１２について説明する。

【００６８】

以下では、まず「システム記述で記述された仕様」の意味内容を説明した上で、「実行制御表形式で記述された仕様」を「システム記述で記述された仕様」に変換するシステム実装変換部１２の処理について説明する。

【００６９】

＜システム記述言語＞

システム記述言語とは、計算機や電子部品回路で実現されるシステムにおける処理実行の動作仕様を記述する言語で、

少なくとも、並列処理、割込み処理、同期処理、繰り返し実行処理、逐次実行処理を記述できる手段を有し、さらにこれらの処理要素を用いた階層関係でシステムの仕様を記述することができる記述言語であり、かつ、

適切な実装変換装置により、計算機や電子部品組込み機器の上で実行可能な形式に変換できるものとする。ここで、実装変換装置とは、例えば、計算機のソフトウェアで実現されるシステムの場合は、コンパイラと呼ばれる変換装置に相当する。

【００７０】

＜ＳｐｅｃＣ言語＞

システム記述言語の代表例としては、仕様記述言語ＳｐｅｃＣが挙げられる。なお、ＳｐｅｃＣについては例えば“SpecC:Specification Language and Methodology”,Daniel D.Gajski,Kluwer Academic Publishers,Dordrecht,ISBN0-7923-7822-9に詳しい。

ＳｐｅｃＣでは、ｐａｒ（並列）、ｆｓｍ（繰返し＋逐次）、ｔｒｙ／ｔｒａｐ／ｉｔｒｐ（割込み），ｎｏｔｉｆｙ／ｗａｉｔ（同期）という特別な言語要素をＣ／Ｃ++に追加した仕様記述言語である（図１０の具体例参照）。

【００７１】

以下の説明では、システム記述言語によって記述された仕様を、ＳｐｅｃＣの言語要素をもとにし、下記に列挙する形式により表現する場合を例にとって説明

する。

【００７２】

・ｐａｒ｛Ａ，Ｂ，Ｃ｝

この言語要素の例では、Ａ，Ｂ，Ｃを並列実行する。

【００７３】

・ｆｓｍ｛｛１，Ａ，goto（２）｝，

｛２，Ｂ，flg==３：goto（１），flg==１：goto（２）｝，

｝

ここで、ｆｓｍの各要素は、

｛ラベル，処理内容，｛条件：条件成立時の遷移｝，…｝、あるいは、

｛ラベル，処理内容，処理終了時の遷移，…｝、あるいは、

｛ラベル，処理内容｝、あるいは、

処理内容

のいずれかで構成され、特に指定が無い場合は、左から順に逐次実行される。すなわち、遷移ラベルの項目が無い場合、｛ラベル，処理内容｝は、処理内容が終了すると、次のｆｍｓ要素を実行することになる。また、ラベル＝１あるいは最左のｆｍｓ要素が最初に実行されるものとする。遷移は、「ｇｏｔｏ（Ｘ）：Ｘはラベル番号」にて表す。例えば、ｇｏｔｏ（１）は、最初のｆｍｓ要素に戻ることを表している。

上記の例では、Ａが実行され、終了すれば、Ｂが実行され、Ｂが終了したとき、変数ｆｌｇの値が３ならばＡを、１ならばＢを実行することを表している。

次に例示するように、ラベルの無い要素は、ラベルを用いた或る実行制御の簡略形であるといえる。

fsm｛A,B,C｝＝fsm｛｛1,A,goto(2)｝，｛2,B,goto(3)｝，｛3,C,…｝，…｝

・ｔｒｙ｛Ａ｝ｔｒａｐ（ｅｖ１）｛Ｂ｝ｉｔｒｐ（ｅｖ２）｛Ｃ｝…

この言語要素の例では、まず、Ａを実行する。そして、Ａの実行中にイベントｅｖ１が発生すれば、Ａを強制終了して、Ｂを実行する。また、Ａの実行中にイベントｅｖ２が発生すれば、Ａを一時停止して、Ｃを実行し、Ｃが終了したとき、Ａの処理を再開する。

なお、ｔｒａｐ（ｅ）｛X｝，ｉｔｒｐ（ｅ２）｛Y｝は、複数あってもよい。

【００７４】

・ｗａｉｔ（ｅｖ）

これは、イベントｅｖの発生を待つ同期処理である。

【００７５】

・ｎｏｔｉｆｙ（ｅｖ）

これは、イベントｅｖを発生させる同期処理である。

【００７６】

・ｆｌｇ＝X

これは、変数ｆｌｇへの値の代入である。

【００７７】

・ｆｌｇ＝＝X

これは、条件判断である。

【００７８】

また、階層構造とは、処理内容がさらにシステム記述言語で書かれた詳細な仕様に展開されることを意味することとする。

階層構造を持つ仕様の例を次に示す。

```
par {
    fsm{A, wait (ev2), B}
    try{C} trap (ev1) {notify (ev2)}
}
```

この仕様例の意味を自然言語で書き下ろすと、次のようになる。

「ＡとＣが最初に並列に実行される。イベントｅｖ１が着たら、Ｃを強制終了し、イベントｅｖ２を発生する。ｅｖ２の発生した時点までにＡが終了していれば、これと同期してＢを起動する。」

＜システム実装変換部１２の処理＞

続いて、システム実装変換部１２の処理について説明する。

【００７９】

前述のように、システム実装変換部１２は、実行制御表形式で記述された仕様を、システム記述言語で記述された仕様に変換する。

【００８０】

図３に、システム実装変換部１２の処理手順の一例を示す。

【００８１】

<<ステップＳ１１：変換ルールによるシステム記述言語要素への展開>>

変換ルールとは、実行制御表におけるプログラム状態遷移単位（１行）を、システム記述言語による仕様に展開するルールである。

【００８２】

システム実装変換部１２は、変換ルールに従って、実行制御表の全てのプログラム状態遷移単位をシステム記述言語の仕様に展開する。

【００８３】

変換ルールの一例としては、例えば「次プログラム」の処理内容

（現プログラムＡ、遷移Ｔ、移動タイプ、終了タイプ、次プログラムＢ）

において、＜起動タイプ、終了タイプ＞の組み合わせに対して、下記に列挙するルールを適用して、システム記述言語で書かれた仕様の部分に展開するものがある。なお、下記において「ｅｖ＿ＸＸ」とあるのは、イベントを表しているものとする。

【００８４】

・＜起動、割込終了＞の場合は、

fsm {wait (ev_A_stop) , B, goto(1)} と

try {A} trap {ev_T} {notify (ev_A_stop) } に展開する。

・＜起動、完了＞の場合は、

fsm {wait (ev_A_end) , B, goto(1)} と

fsm {A, notify (ev_A_end) , goto(1)} に展開する。

・＜起動、割込一時停止＞の場合は、

fsm {wait (ev_A_interrupt) , B, goto(1)} と

try {A} itrp {ev_T} {fsm {n tify (ev_A_interrupt) , wait (ev_A_resume) } } に展開する。

・＜復帰、完了＞の場合は、

fsm {wait (ev_A_end) , notify (ev_B_resume) , goto(1)} と

fsm {A, notify (ev_A_end) , goto(1)} に展開する。

・＜復帰、割込終了＞の場合は、

fsm {wait (ev_A_stop_resume) , notify (ev_B_resume) , goto(1)} と

try {A} itrp {ev_T} {notify (ev_A_stop_resume) } に展開する。

【００８５】

システム実装変換部１２は、これらのような展開を、実行制御表に含まれる全てのプログラム状態遷移単位に対して実行する。

【００８６】

<<ステップＳ１２：処理内容単位の仕様としてまとめる>>

展開された仕様の一部の集合を、その個々の処理内容（Ａ，Ｂ，…）を中心にして次の形式にまとめる。

ｆｍｓ｛開始待ち、処理内容、完了後処理、遷移｝、あるいは、

ｆｍｓ｛開始待ち、処理内容、遷移｝

ここで、開始待ちについては、

開始待ち＝par｛wait（同期イベント１），wait（同期イベント２）｝、あるいは、

開始待ち＝wait（同期イベント１，同期イベント２）

の形式をとるものとする。

また、処理内容については、

処理内容＝処理内容Ａ，Ｂ，…、あるいは、

処理内容＝try｛Ａ｝trap（遷移イベント）｛遷移後処理｝と

try｛Ａ｝itrp（遷移イベント）｛遷移後処理｝とから

構成されるリスト

の形式をとるものとする。

また、完了後処理については、

完了後処理＝ｎｏｔｉｆｙ（同期イベント）

の形式をとるものとする。

また、遷移については、

遷移＝ｇｏｔｏ（Ｘ），Ｘはラベル

の形式をとるものとする。

【００８７】

ここで説明のために変換ルールにて展開された仕様記述を以下のように分類する。

（１）開始のｆｓｍ

開始のｆｓｍとは、処理内容Ｂに対して、

ｆｓｍ｛ｗａｉｔ（Ｘ），Ｂ，ｇｏｔｏ（１）｝

の形式を持つ要素を呼ぶこととする。

【００８８】

（２）終了のｔｒｙ

終了のｔｒｙとは、処理内容Ｘに対して、

ｔｒｙ｛Ｘ｝ｔｒａｐ（ｅ）｛Ｙ｝

ｔｒｙ｛Ｘ｝ｉｔｒｐ（ｅ）｛Ｙ｝

の形式を持つ要素を呼ぶこととする。

（３）終了のｆｓｍ

終了のｆｓｍとは、処理内容Ｘに対して、

ｆｓｍ｛Ｘ，ｎｏｔｉｆｙ（ｅ），ｇｏｔｏ（１）｝

の形式を持つ要素を呼ぶこととする。

【００８９】

以下に、仕様統合のルールの例を列挙する。

・『開始のｆｓｍと終了のｆｓｍの統合』

ｆｓｍ｛ｗａｉｔ（ｅ），Ａ，ｇｏｔｏ（１）｝

ｆｓｍ｛Ａ，ｎｏｔｉｆｙ（ｆ），ｇｏｔｏ（１）｝

を、

ｆｓｍ｛ｗａｉｔ（ｅ），Ａ，ｎｏｔｉｆｙ（ｆ），ｇｏｔｏ（１）｝

として統合する。

・『開始のｆｓｍ同士の統合』

ｆｓｍ｛ｗａｉｔ（Ａ），Ｂ，ｇｏｔｏ（１）｝

ｆｓｍ｛ｗａｉｔ（Ｃ），Ｂ，ｇｏｔｏ（１）｝

を、

ｆｓｍ｛ｗａｉｔ（Ａ，Ｃ），Ｂ，ｇｏｔｏ（１）｝

として統合する。

開始にｗａｉｔが複数ある場合には、これの論理和（少なくとも１つのイベントを持つ）に統合する。

【００９０】

・『終了のｔｒｙ同士の統合』

ｔｒｙ｛Ａ｝ｔｒａｐ（ｅ１）｛Ｃ｝

ｔｒｙ｛Ａ｝ｔｒａｐ（ｅ２）｛Ｄ｝

ｔｒｙ｛Ａ｝ｔｒａｐ（ｅ３）｛Ｅ｝

を、

ｔｒｙ｛Ａ｝ｔｒａｐ（ｅ１）｛Ｃ｝ｔｒａｐ（ｅ２）｛Ｄ｝ｔｒａｐ（ｅ３）｛Ｅ｝

に統合する。

・『開始のｆｓｍと終了のｔｒｙの統合』

ｆｓｍ｛ｗａｉｔ（Ａ），Ｂ，ｇｏｔｏ（１）｝

ｔｒｙ｛Ｂ｝ｔｒａｐ（ｅ）｛Ｃ｝

を、

ｆｓｍ｛ｗａｉｔ（Ａ），ｔｒｙ｛Ｂ｝ｔｒａｐ（ｅ）｛Ｃ｝，ｇｏｔｏ（１）｝

に統合する。同様に、

ｆｓｍ｛ｗａｉｔ（Ａ），Ｂ，ｇｏｔｏ（１）｝

ｔｒｙ｛Ｂ｝ｉｔｒｐ（ｅ）｛Ｃ｝

を、

ｆｓｍ｛ｗａｉｔ（Ａ），ｔｒｙ｛Ｂ｝ｉｔｒｐ（ｅ）｛Ｃ｝，ｇｏｔｏ（１）｝

に統合する。

・『終了のｆｓｍと終了のｔｒｙの統合』

```
ｆｓｍ｛A，ｎｏｔｉｆｙ（ｅｎｄ），ｇｏｔｏ（1）｝
ｔｒｙ｛B｝ｔｒａｐ（ｅ）｛C｝
```

を、

```
ｆｓｍ｛ｔｒｙ｛ｆｓｍ｛A，ｎｏｔｉｆｙ（ＦＩＮＩＳＨ）｝｝
          ｔｒａｐ（ｅ）｛C｝
          ｔｒａｐ（ＦＩＮＩＳＨ）｛ｎｏｔｉｆｙ（ｅｎｄ）｝，
    ｇｏｔｏ（1）｝
```

に統合する。ここで、ＦＩＮＩＳＨは、Ａの終了を知らせるイベントであり、これをｔｒａｐ（ＦＩＮＩＳＨ）にて検知して、最終的にＡの終了イベントｅｎｄを発行する。

【００９１】

<<ステップＳ１３：全体のシステム仕様としてまとまる>>

ステップＳ１２によって得られた仕様の一部を、並列実行構造（ｐａｒ）にて束ねると、最終的なシステム仕様が完成する。

```
システム仕様＝
    ｐａｒ｛
        ｆｓｍ｛開始待ち，処理内容Ａ，完了後処理｝，
        ｆｓｍ｛開始待ち，処理内容Ｂ，完了後処理｝，
        ｆｓｍ｛開始待ち，処理内容Ｃ，完了後処理｝，
                        ：
    ｝
```

以上の手順により、実行制御表形式で記述された仕様がシステム記述言語で記述された仕様に変換される。

【００９２】

なお、この手順で用いた変換ルールは一例であり、生成されるシステム記述言語の最適化（例えば記述の大きさを最小にする）や実際の計算や電子機器による実装の容易性のために、他の変換ルールを用いることもできる。

【００９３】

また、得られたシステム仕様についても、機能を保存したまま、変換を行い最適化を行う等の手段を用いて、所望の構造に変換することもできる。

【００９４】

さて、以下では、本実施形態について、Ｃｏｍｐａｃｔ　Ｄｉｓｋの演奏装置（ＣＤ　ｐｌａｙｅｒ）の設計を例とって、さらに詳しく説明する。ここでのＣＤ　ｐｌａｙｅｒとは、例えば、制御ボタン（制御ボタンはハードウェアのボタンでもＧＵＩ画面上のボタンでもよい）として、ｐｌａｙ（演奏）、ｐａｕｓｅ（一時停止）、ｓｔｏｐ（停止）を押して、演奏機械を制御する装置である。なお、以下では、＜ＸＸＸ＞は、制御ボタンＸＸＸが押されたイベントを意味するものとする。

【００９５】

ここで、ＣＤ　ｐｌａｙｅｒは、その状態として、ｓｔｏｐｐｅｄ（停止中）、ｐｌａｙｉｎｇ（演奏中）、ｐａｕｓｉｎｇ（一時停止中）の３つの状態を持つものとし、制御ボタンの押し下げをイベントとする状態遷移表形式でＣＤ　ｐｌａｙｅｒの仕様を記述する。図４に、この場合の状態遷移表の一例を示す。

【００９６】

状態遷移表の各行は、現状態に対してイベントが発生したとき、どの状態に遷移するかを記述している。また、実行制御の項目の細目に、ｓｔａｒｔ，ｓｔｏｐ，ｉｎｔｅｒｒｕｐｔ，ｒｅｓｕｍｅがあり、それぞれ、遷移時に、起動（ｓｔａｒｔ）、割込終了（ｓｔｏｐ）、割込一時停止（ｉｎｔｅｒｒｕｐｔ）、割込一時停止からの復帰（ｒｅｓｕｍｅ）を行う、処理内容の単位（プログラム）が定義してある。ＣＤ　ｐｌａｙｅｒにおける処理内容の単位は、ｐｌａｙ（演奏実行）、ｐａｕｓｅ（一時停止実行）の２種類である。

【００９７】

また、終了（ｆｉｎｉｓｈ）は、実行中の処理内容が終了したことをイベントとして利用することを示し、例えば、遷移番号［３］では、ｐｌａｙが終了したことがｐｌａｙｉｎｇからｓｔｏｐｐｅｄに遷移するイベントであると定義している。このとき、イベント欄には、［ＦＩＮＩＳＨ］と記述していて、終了のイ

ベントであることを明示している。

【００９８】

また、ｓｔａｒｔ状態からの遷移（遷移［０］）の場合は、開始プログラム状態（ｓｔａｒｔ）を強制終了するものとみなす。

【００９９】

＜実行制御への変換＞

実行制御表変換部１１により状態遷移表を実行制御表に変換する。

【０１００】

まず、図２のステップＳ１の手順に従い、状態遷移表の実行制御項目において、開始に関する項目（ｓｔａｒｔ，ｒｅｓｕｍｅ）と終了に関する項目（ｓｔｏｐ，ｆｉｎｉｓｈ，ｉｎｔｅｒｒｕｐｔ）両方に処理内容（プログラム）がある状態遷移単位を選択し、これを＜現プログラム、次プログラム＞の組み合わせとして展開する。

【０１０１】

開始に関する項目と終了に関する項目とを両方持っているものは、遷移［２］，［４］，［５］であるから、最初に展開されるプログラム状態遷移単位は、図５のようになる。

【０１０２】

次に、図２のステップＳ２の手順に従い、遷移の接続関係を辿ると、「次プログラム」が無い遷移［７］，［３］，［６］，［０］に対しては、それぞれ遷移の接続の順方向を参照する。例えば遷移［７］の場合は、ｓｔｏｐｐｅｄ状態を介して遷移［１］と順方向に接続しており、遷移［１］には、「次プログラム」があるので、図６に示す新たなプログラム状態遷移を得る。遷移［３］，［６］，［０］の場合も同様に、図６に示す新たなプログラム遷移状態を得る。

【０１０３】

ここで、図２のステップＳ３の手順に従い、遷移［０］，［３］，［６］，［７］と［１］の間に新しい処理内容（Ｘｓｔｏｐ）を挿入することにする。

【０１０４】

開始と終了を考慮して、

状態遷移表における遷移［１］のｓｔａｒｔ項目

状態遷移表における遷移［７］，［３］，［６］，［０］のｓｔｏｐ項目に挿入する。

【０１０５】

図４の状態遷移表にＸｓｔｏｐが挿入されたもの（最終的に得られる実行制御表に対応する状態遷移表）を図７に、最終的に得られる実行制御表を図８に示す。

【０１０６】

＜システム仕様への変換＞

システム実装変換部１２により実行制御表形式の仕様をシステム記述言語形式の仕様に変換する。

【０１０７】

まず、図３のステップＳ１１の手順と、変換ルールに従い、各々のプログラム状態遷移単位をＳｐｅｃＣ形式に変換する。この結果を、図９に示す。

【０１０８】

これらを処理内容（ｐｌａｙ，ｐａｕｓｅ，Ｘｓｔｏｐ）毎にまとめると、次のようになる。

【０１０９】

```
[play]
    fsm {wait (ev_Xstop_stop) , play, goto(1)}
    fsm {play, notify (ev_play_end) , goto(1)}
    try {play} itrp (ev_[2])  {fsm {notify (ev_play_interrupt) ,
                              wait (ev_play_resume) } }
    try {play} trap (ev_[6])  {notify (ev_play_stop) }
[pause]
    fsm {wait (ev_play_interrupt) , pause, goto(1)}
    try {pause} trap (ev_[5])  {notify (ev_pause_st p) }
    try {pause} trap (ev_[7])  {notify (ev_pause_st p) }
    try {pause} trap (ev_[4])  {n tify (ev_pause_st p_resume) }
```

[Ｘｓｔｏｐ]

fsm {wait (start) , Xstop, g to(1)}

fsm {wait (ev_play_end) , Xstop, goto(1)}

fsm {wait (ev_play_stop) , Xstop, goto(1)}

fsm {wait (ev_pause_stop) , Xstop, goto(1)}

try {Xstop} trap (ev_[1]) {notify (ev_Xstop_stop) }

[ＯＴＨＥＲＳ]

fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto(1)}

fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto(1)}

次に、図３のステップＳ１２の手順と仕様統合ルールに従って、統合を行う。この結果を、以下に示す。

【０１１０】

[Ｘｓｔｏｐ]

fsm {wait (start) , Xstop, goto (1) }

fsm {wait (ev_play_end) , Xstop, goto (1) }

fsm {wait (ev_play_stop) , Xstop, goto (1) }

fsm {wait (ev_pause_stop) , Xstop, goto (1) }

try {Xstop} trap (ev_ [1]) {notify (ev_Xstop_stop) }

は、開始のｆｓｍを統合して、

fsm {wait (start, ev_play_end, ev_play_stop, ev_pause_stop) , Xstop, goto (1) }

try {Xstop} trap (ev_ [1]) {notify (ev_Xstop_stop) }

とし、開始のｆｓｍと終了のｔｒｙを統合して、

fsm {wait (start, ev_play_end, ev_play_stop, ev_pause_stop) ,

try {Xstop} trap (ev_ [1]) {notify (ev_Xstop_stop) } ,

got (1)

}

とする。

【０１１１】

[ｐｌａｙ]

```
fsm {wait (ev_Xstop_stop) , play, goto (1) }
fsm {play, notify (ev_play_end) , goto (1) }
try {play}  (ev_ [2] ) itrp {fsm {notify (ev_play_intrrupt) ,
                              wait (ev_play_resume) } }
try {play}  (ev_ [6] ) trap {notify (ev_play_stop) }
```

は、終了のｔｒｙを統合し終了のｆｓｍをｔｒｙに統合すると、

```
fsm {wait (ev_Xstop_stop) , play, goto (1) }
fsm {
    try {fsm {play, notify (FINISH) } }
        itrp (ev_ [2] )  {fsm {notify (ev_play_intrrupt) , wait (e
v_play_resume) } }
        trap (ev_ [6] )  {notify (ev_play_stop) }
        trap (FINISH)  {notify (ev_play_end) } ,
    goto (1)
}
```

とする。さらに、開始のｆｓｍと終了のｆｓｍを統合して、

```
fsm {wait (ev_Xstop_stop) ,
    try {fsm {play, notify (FINISH) } }
        itrp (ev_ [2] )  {fsm {notify (ev_play_intrrupt) , wait (e
v_play_resume) } }
        trap (ev_ [6] )  {notify (ev_play_stop) }
        trap (FINISH)  {notify (ev_play_end) } ,
    goto (1)
}
```

とする。

【０１１２】

[ｐａｕｓｅ]

```
fsm {wait (ev_play_interrupt) , pause, goto (1) }
try {pause} trap (ev_ [5] )  {notify (ev_pause_stop_resume) }
try {pause} trap (ev_ [7] )  {notify (ev_pause_stop) }
try {pause} trap (ev_ [4] )  {notify (ev_pause_stop_resume) }
```

は、ｔｒｙを統合して、

```
fsm {wait (ev_play_interrupt) , pause, goto (1) }
try {pause} trap (ev_ [5] )  {notify (ev_pause_stop_resume) }
        trap (ev_ [7] )  {notify (ev_pause_stop) }
        trap (ev_ [4] )  {notify (ev_pause_stop_resume) }
```

とし、さらに起動のｆｓｍと終了のｔｒｙを統合して、

```
fsm {wait (ev_play_interrupt) ,
  try {pause} trap (ev_ [5] )  {notify (ev_pause_stop_resume) }
        trap (ev_ [7] )  {notify (ev_pause_stop) }
        trap (ev_ [4] )  {notify (ev_pause_stop_resume) } ,
  goto (1)
}
```

とする。

【０１１３】

[ＯＴＨＥＲＳ]

同じものは1つに統合して、

```
fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto
(1) }
```

とする。

【０１１４】

最終的に得られる、システム記述言語形式の仕様は、図１０のようになる。

【０１１５】

なお、以上の各機能は、ソフトウェアとして実現可能である。

また、本実施形態は、コンピュータに所定の手段を実行させるための（あるい

はコンピュータを所定の手段として機能させるための、あるいはコンピュータに所定の機能を実現させるための）プログラムとして実施することもでき、該プログラムを記録したコンピュータ読取り可能な記録媒体として実施することもできる。

【０１１６】

なお、この発明の実施の形態で例示した構成は一例であって、それ以外の構成を排除する趣旨のものではなく、例示した構成の一部を他のもので置き換えたり、例示した構成の一部を省いたり、例示した構成に別の機能あるいは要素を付加したり、それらを組み合わせたりすることなどによって得られる別の構成も可能である。また、例示した構成と論理的に等価な別の構成、例示した構成と論理的に等価な部分を含む別の構成、例示した構成の要部と論理的に等価な別の構成なども可能である。また、例示した構成と同一もしくは類似の目的を達成する別の構成、例示した構成と同一もしくは類似の効果を奏する別の構成なども可能である。

また、この発明の実施の形態で例示した各種構成部分についての各種バリエーションは、適宜組み合わせて実施することが可能である。

また、この発明の実施の形態は、個別装置としての発明、関連を持つ２以上の装置についての発明、システム全体としての発明、個別装置内部の構成部分についての発明、またはそれらに対応する方法の発明等、種々の観点、段階、概念またはカテゴリに係る発明を包含・内在するものである。

従って、この発明の実施の形態に開示した内容からは、例示した構成に限定されることなく発明を抽出することができるものである。

【０１１７】

本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【０１１８】

【発明の効果】

本発明によれば、状態遷移表形式に処理内容（プログラム）を直接入力して、これを実行可能なシステムに変換するシステム生成手段を具備しているので、状

態遷移表の柔軟さを損なずにシステム設計を行うことができ、また処理内容の実行制御命令自体は瞬間に実行されるので意味論的に状態遷移の仕組みと整合性が取れたシステム設計を行うことができる。これによって、システム分析から設計までを一環して支援し、システム分析結果に対応した精度よいシステム設計の支援を行って、システム開発期間を短縮させ、設計品質を向上させることができる。

【図面の簡単な説明】

【図１】

本発明の実施の形態に係る設計支援装置の構成例を示す図

【図２】

同実施形態に係る設計支援装置の実行制御表変換部の処理手順の一例を示すフローチャート

【図３】

同実施形態に係る設計支援装置のシステム実装変換部の処理手順の一例を示すフローチャート

【図４】

状態遷移表形式の仕様の一例を示す図

【図５】

状態遷移表形式の仕様から実行制御表形式の仕様への変換について説明するための図

【図６】

状態遷移表形式の仕様から実行制御表形式の仕様への変換について説明するための図

【図７】

状態遷移表形式の仕様の一例を示す図

【図８】

実行制御表形式の仕様の一例を示す図

【図９】

実行制御表形式の仕様からシステム記述言語形式の仕様への変換について説明

するための図

【図１０】

システム記述言語形式の仕様の一例を示す図

【図１１】

同実施形態に係るシステム設計支援装置の構成例を示す

【符号の説明】

１…システム生成部

２…入出力部

３…通信部

４…プログラム実行部

１１…実行制御表変換部

１２…システム実装変換部

１００…システム設計支援装置

１０１…仕様モデル記述部

１０２…アーキテクチャ探索部

１０３…コミュニケーション合成部

１０４…ハードウェア仕様生成部

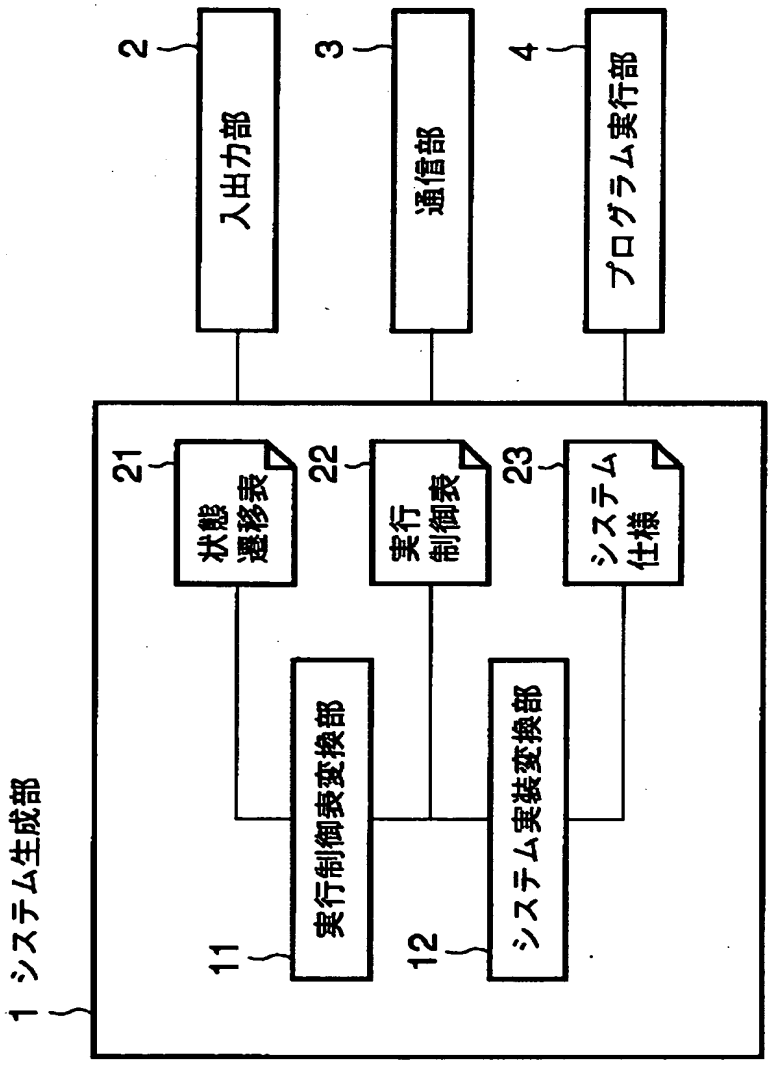
１０５…部品化・再利用部

１０６…ソフトウェア仕様生成部

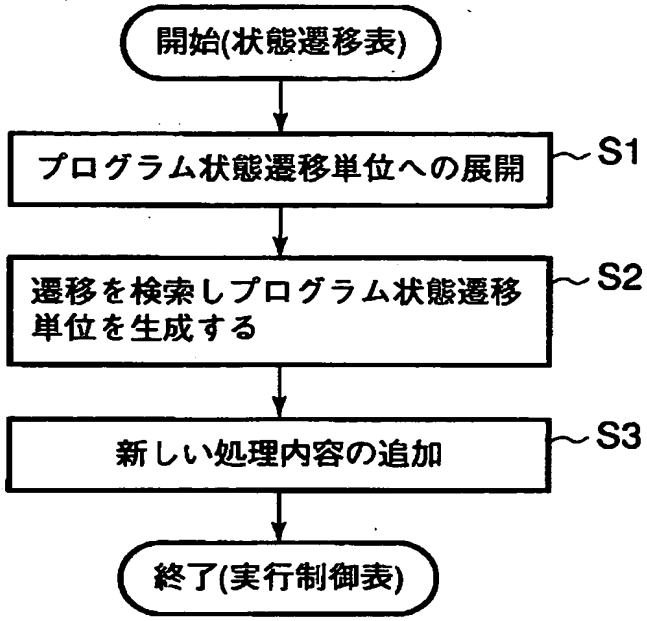
１０７…システム仕様記録部

【書類名】　　図面

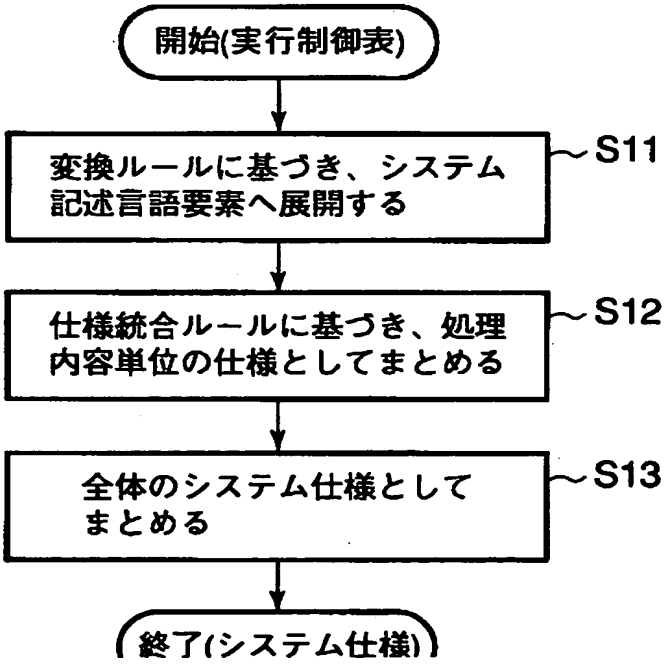
【図１】


1 システム生成部
2
入出力部
3
通信部
4
プログラム実行部
21
状態遷移表
22
実行制御表
23
システム仕様
11
実行制御表変換部
12
システム実装変換部

【図２】


開始(状態遷移表)
プログラム状態遷移単位への展開
S1
遷移を検索しプログラム状態遷移単位を生成する
S2
新しい処理内容の追加
S3
終了(実行制御表)

【図３】


開始(実行制御表)
変換ルールに基づき、システム記述言語要素へ展開する
S11
仕様統合ルールに基づき、処理内容単位の仕様としてまとめる
S12
全体のシステム仕様としてまとめる
S13
終了(システム仕様)

【図４】

番号	現状態	イベント	条件	次状態	実行制御				
					start	stop	interrupt	resume	finish
[0]	start			stopped	START				
[1]	stopped	<play>		playing	play				
[2]	playing	<pause>		pausing	pause		play		
[3]	playing	[FINISH]		stopped					play
[4]	pausing	<play>		playing		pause		play	
[5]	pausing	<pause>		playing		pause		play	
[6]	playing	<stop>		stopped		play			
[7]	pausing	<stop>		stopped		pause			

【図５】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
	play	[２]	起動	割込一時停止	pause
	pause	[４]	復帰	割込終了	play
	pause	[５]	復帰	割込終了	play
	START	[０]		割込終了	
		[１]	起動		play
	play	[３]		完了	
	play	[６]		割込終了	
	pause	[７]		割込終了	

【図６】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
	pause	[７]→[１]	起動	割込終了	play
	play	[６]→[１]	起動	割込終了	play
	play	[３]→[１]	起動	割込終了	play
	START	[０]→[１]	起動	完了	play

【図７】

番号	現状態	イベント	条件	次状態	実行制御				
					start	stop	Interrupt	resume	finish
[0]	start			stopped	Xstop				
[1]	stopped	<play>		playing	play	Xstop			
[2]	playing	<pause>		pausing	pause		play		
[3]	playing	[FINISH]		stopped	Xstop				play
[4]	pausing	<play>		playing		pause		play	
[5]	pausing	<pause>		playing		pause		play	
[6]	playing	<stop>		stopped	Xstop	play			
[7]	pausing	<stop>		stopped	Xstop	pause			

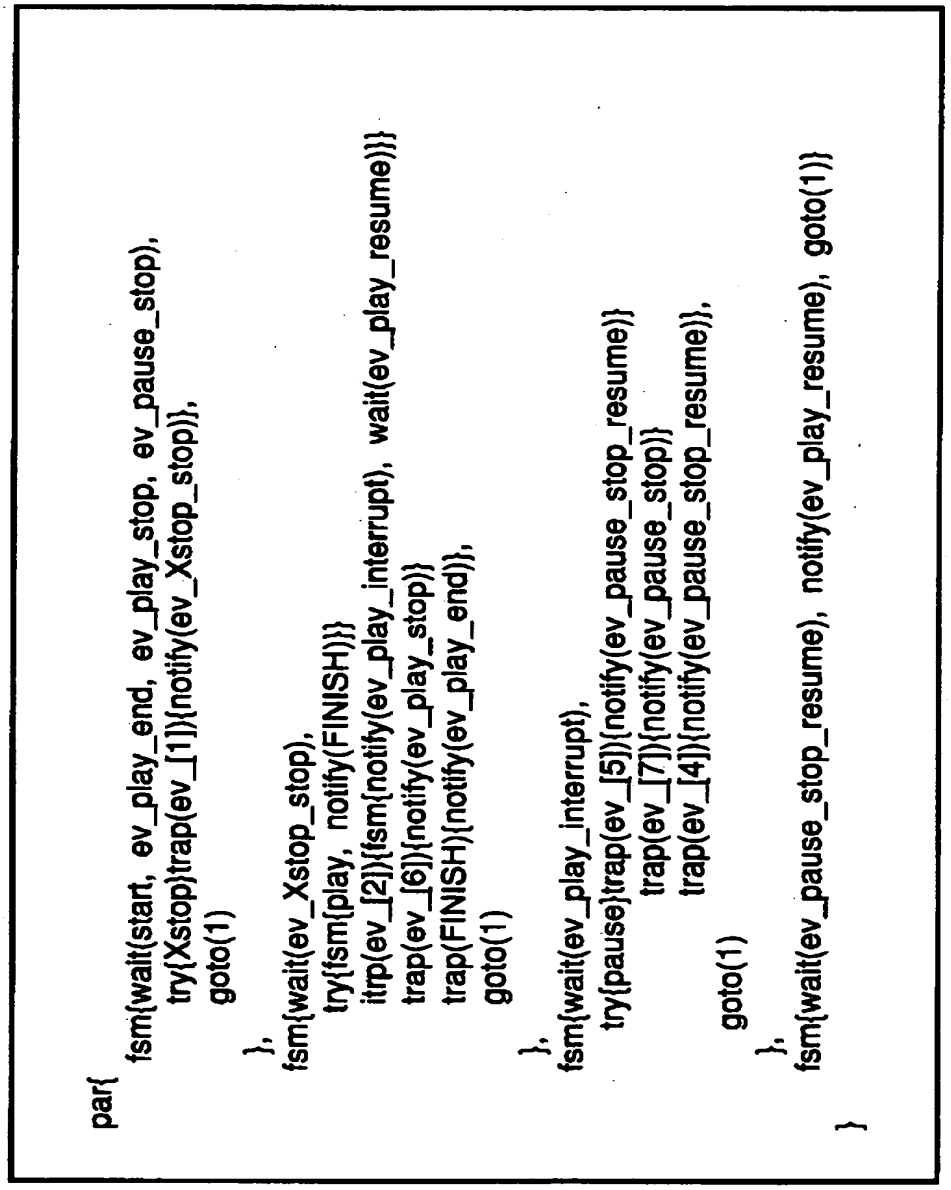
【図８】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
(0)	START	[0]	起動		Xstop
(1)	Xstop	[1]	起動	割込終了	play
(2)	play	[2]	起動	割込一時停止	pause
(3)	play	[3]	復帰	完了	Xstop
(4)	pause	[4]	復帰	割込終了	play
(5)	pause	[5]	復帰	割込終了	play
(6)	play	[6]	起動	割込終了	Xstop
(7)	pause	[7]	起動	割込終了	Xstop

【図９】

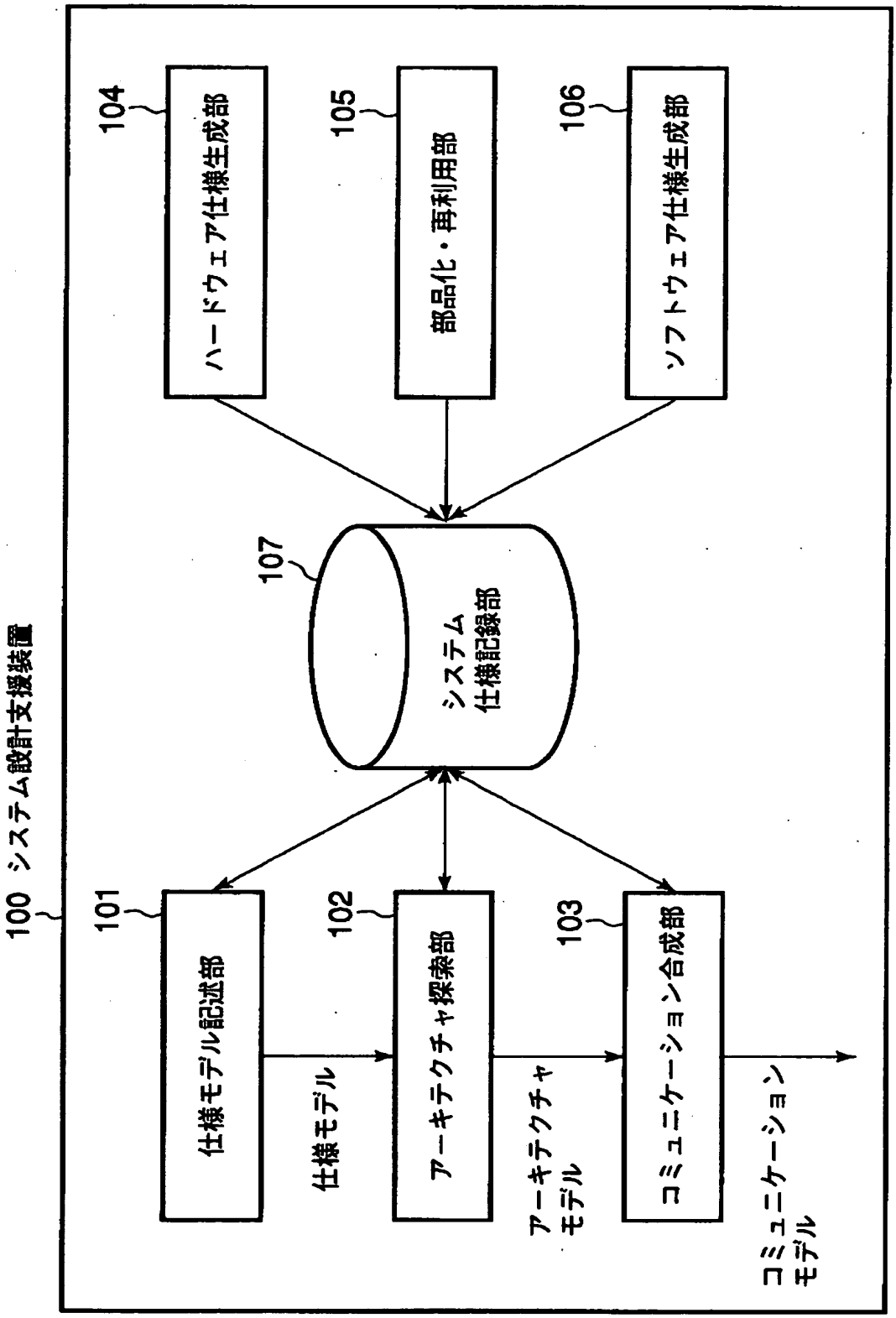
番号	現ﾌﾟﾛｸﾞﾗﾑ	遷移	起動ﾀｲﾌﾟ	終了ﾀｲﾌﾟ	次ﾌﾟﾛｸﾞﾗﾑ	変換結果
(０)	START	[０]	起動		Xstop	fsm{wait(start), Xstop, goto(1)}
(１)	Xstop	[１]	起動	割込終了	play	fsm{wait(ev_Xstop), play, goto(1)} try{Xstop}trap(ev_[1]){notify(ev_Xstop_stop)}
(２)	play	[２]	起動	割込 一時停止	pause	fsm{wait(ev_play_interrupt), pause, goto(1)} try{play}itrp(ev_[2]){fsm{notify(ev_play_interrupt), wait(ev_play_resume)}}
(３)	play	[３]	起動	完了	Xstop	fsm{wait(ev_play_end), Xstop, goto(1)} fsm{play, notify(ev_play_end), goto(1)}
(４)	pause	[４]	復帰	割込終了	play	fsm{wait(ev_pause_stop_resume), notify(ev_play_resume), goto(1)} try{pause}trap(ev_[4]){notify(ev_pause_stop_resume)}
(５)	pause	[５]	復帰	割込終了	play	fsm{wait(ev_pause_stop), notify(ev_play_resume), goto(1)} try{pause}trap(ev_[5]){notify(ev_pause_stop)}
(６)	play	[６]	起動	割込終了	Xstop	fsm{wait(ev_play_stop), Xstop, goto(1)} try{play}trap(ev_[6]){notify(ev_play_stop)}
(７)	pause	[７]	起動	割込終了	Xstop	fsm{wait(ev_pause_stop), Xstop, goto(1)} try{pause}trap(ev_[7]){notify(ev_pause_stop)}

【図１０】



```
par{
    fsm{wait(start, ev_play_end, ev_play_stop, ev_pause_stop),
        try{Xstop}trap(ev_[1]){notify(ev_Xstop_stop)},
            goto(1)
    },
    fsm{wait(ev_Xstop_stop),
        try{fsm{play, notify(FINISH)}}
        itrp(ev_[2]){fsm{notify(ev_play_interrupt), wait(ev_play_resume)}}
        trap(ev_[6]){notify(ev_play_stop)}
        trap(FINISH){notify(ev_play_end)},
            goto(1)
    },
    fsm{wait(ev_play_interrupt),
        try{pause}trap(ev_[5]){notify(ev_pause_stop_resume)}
                trap(ev_[7]){notify(ev_pause_stop)}
                    trap(ev_[4]){notify(ev_pause_stop_resume)},
        goto(1)
    },
    fsm{wait(ev_pause_stop_resume), notify(ev_play_resume), goto(1)}
}
```

【図１１】


100 システム設計支援装置
101
仕様モデル記述部
102
アーキテクチャ探索部
103
コミュニケーション合成部
104
ハードウェア仕様生成部
105
部品化・再利用部
106
ソフトウェア仕様生成部
107
システム
仕様記録部
仕様モデル
アーキテクチャ
モデル
コミュニケーション
モデル

【書類名】　　要約書

【要約】

【課題】　システム分析結果に対応した精度良いシステム設計の支援を行って開発期間を短縮し設計品質を向上できる設計支援装置を提供すること。

【解決手段】　状態遷移表形式のシステム仕様（２１）を、現状態、イベント、次状態、実行制御に関する情報を含む状態遷移単位情報の集合で構成する。実行制御に関する情報は、当該状態遷移に関連して実行制御されるプログラム及びその実行制御内容（起動、割込終了、割込一時停止、終了、復帰）を含む。実行制御表変換部１１は、この状態遷移表に基づいて、実行制御表形式のシステム仕様（２２）を生成する。実行制御表は、遷移、現実行処理内容、その終了タイプ、次プログラム、その起動タイプを含むプログラム状態遷移単位情報の集合からなる。システム実装変換部１２は、実行制御表に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様（２３）を生成する。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００３０７８〕

１．変更年月日　　１９９０年　８月２２日
［変更理由］　新規登録
住　所　神奈川県川崎市幸区堀川町７２番地
氏　名　株式会社東芝

２．変更年月日　　２００１年　７月　２日
［変更理由］　住所変更
住　所　東京都港区芝浦一丁目１番１号
氏　名　株式会社東芝

出証特２００１－３１０８８９５